XML

CSC480: Semantic Web Technologies

Dr. Lisa Frye
frye@kutztown.edu
Kutztown University

Background

- eXtensible Markup Language

- Transports data
- No predefined tags

Benefits

- Self-describing data
- Rules to limit the type of data for elements
- Custom data structures
- Allows for presentation styles
- Rich set of tools for linking
- Interchange data between proprietary formats
- Tools to define a standard syntax
- Robust and reliable data searching capabilities

Syntax Rules

- Root element
- Opening (start) and closing (end) tag
- Case sensitive
- Elements must be properly nested
- Attribute values must be quoted
- Attributes must be in opening (start) tag
- Attributes cannot appear more than once

XML Document Structure

- Physical - content
- Logical - organization

- Provide Structure
  - DTDs
  - XML Schema

Logical Structure

- Prolog
  - Declaration statement
  - External structure document references (optional)
- Elements
- Epilog (optional)

Declaration Statement

- Must be first row, first position
- Attributes
  - Version
  - Encoding
  - Standalone
- <?xml version=”1.0” encoding=”UTF-16”?>

Processing Instruction (PI)

- Pass instructions to application
- <? target instruction ?>
- Ignored by XML parser
- Primary use – link to stylesheets
  - <?stylesheet type="text/css" href="mystyle.css"?>

Namespaces

- Name clash
- Namespace declaration – xmlns
- Default DTD
  - xmlns=”location”
- Specify prefix
  - prefix**:**name
  - xmlns:prefix=”location”

Namespace Example

- <xmlns:employee=http://faculty.kutztown.edu/XML/employee>
- <employee:name>Lisa Frye</employee:name>

XML Document Contents

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

Elements

- Syntax
  - Opening (begin) tag
  - Contents
  - Closing (end) tag
- Root element

Naming Rules

- Can contain letters, numbers, and other characters
- Cannot start with a number or punctuation character
- Cannot start with the letters xml (or XML, or Xml, etc)
- Cannot contain spaces

Element Example

```
<family_tree>
        <mother>Sally</mother>
        <father>Joe</father>
        <children>
                <child>Larry</child>
                <child>Larry</child>
                <child>Larry</child>
        </children>
</family_tree>
```

Attributes

- Name-value pairs
- Syntax
  - name=value
- Nested element vs. attribute

Attribute Example

```
<property>
        <address>123 Main Street</address>
        <city>Kutztown</city>
        <state>PA</state>
        <zip>19530</zip>
</property>

<property  address=”123 Main Street”  city=”Kutztown”
                        state=”PA” zip=”19530”/>
```

Entities

- Placeholders for content
- Content
  - Special characters
  - Text
  - XML markup
  - Binary data

Content Entities

- Replace content throughout XML document
- Program constants
- Defined in internal DTD
- Syntax
  - <!ENTITY entity-name "entity-value">

Content Entity DTD

```
<!DOCTYPE business [
        <!ENTITY name “ACME Shipping Company”>
        <!ENTITY address “123 Main Street”>
        <!ENTITY city “Kutztown”>
        <!ENTITY state “PA”>
        <!ENTITY zip “19530”>
]>
```

Content Entity XML

```
<business>
        <bus_name>Business Name: &name;</bus_name>
        <bus_addr>Business Address: &address;</bus_addr>
        <bus_city>Business Address: &city;</bus_city>
        <bus_state>Business Address: &state;</bus_state>
        <bus_zip>Business Address: &zip;</bus_zip>
        <ship_addr>Business Address: &address;</ship_addr>
        <ship_city>Business Address: &city;</ship_city>
        <ship_ state>Business Address: &state;</ship_state>
        <ship_zip>Business Address: &zip;</ship_zip>
</business>
```

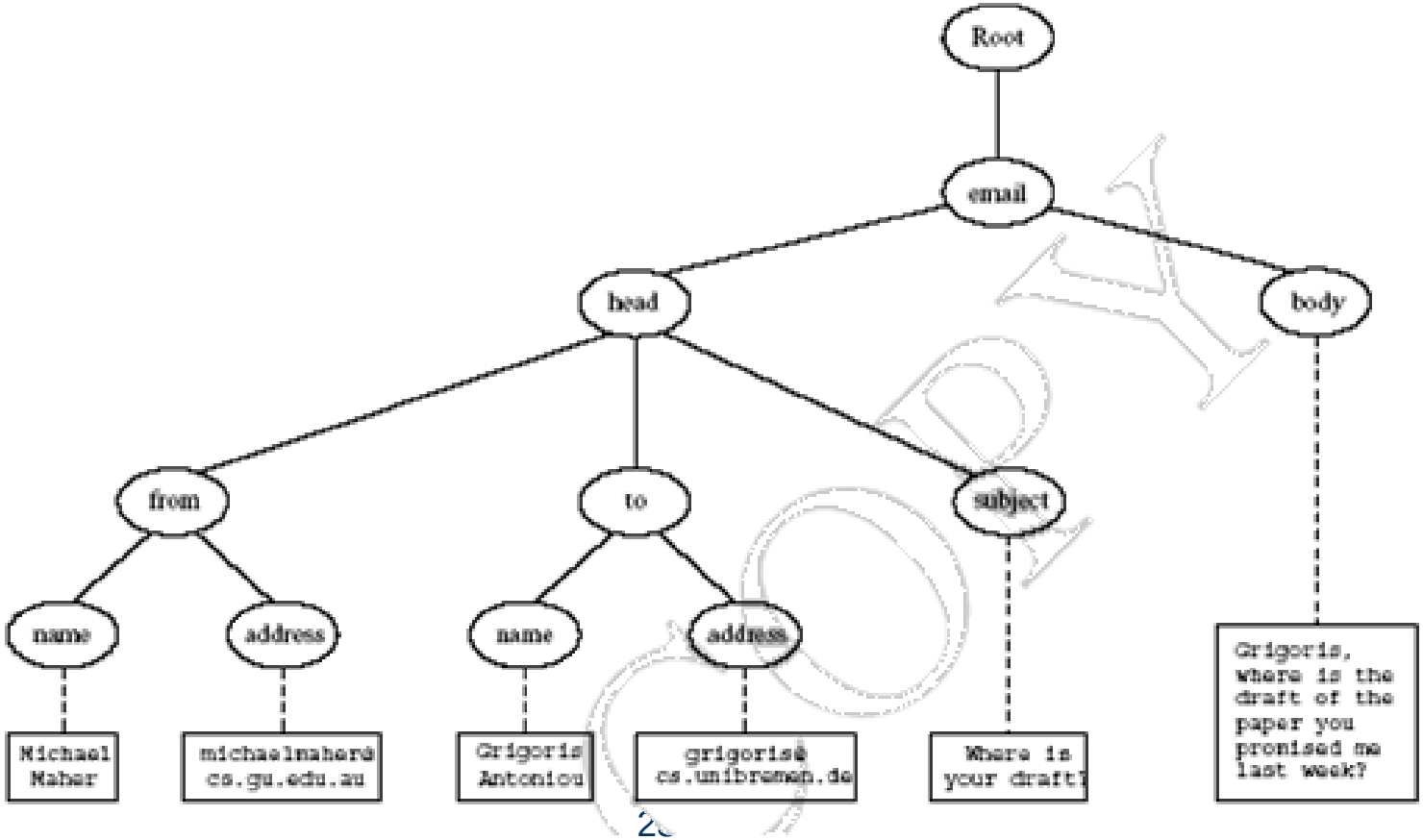
PCDATA and CDATA

- Parsed character data
  - Content that is parsed by XML parser
  - Examined for entities and markup
- Character data
  - Content ignored by XML parser
  - Used when data contains special characters
  - <![CDATA[<< text >>]]>

The Tree Model - An Example

```
<email>
    <head>
        <from name="Michael Maher"
              address="michaelmaher@cs.gu.edu.au"/>
        <to name="Grigoris Antoniou"
              address="grigoris@cs.unibremen.de"/>
        <subject>Where is your draft?</subject>
    </head>
    <body>
        Grigoris, where is the draft of the paper you promised me
        last week?
    </body>
</email>
```

The Tree Model - An Example (2)



Tree Model Example

- http://faculty.kutztown.edu/frye/secure/CSC480SW/Examples/bookstore.xml

- Root element?
- Other elements?
- Attributes?
- Draw the tree

Document Type Definitions (DTDs)

- Define structure of XML documents
- Internal or external
- Syntax
  - <!DOCTYPE [ ]>
- Elements
  - <!ELEMENT element-name category>
  - <!ELEMENT element-name (element-content)>

Content Model Types

Content Model	Declaration Syntax	Description
Text	(#PCDATA)	Text or character data (no elements)
Elements	(element_name)	Other elements
Mixed content	(#PCDATA, element_name)	Contains both text and other elements. The #PCDATA declaration must appear first
Empty	EMPTY	Contains no content
ANY	ANY	Can contain text or elements

Element Examples

- Empty elements
  - <!ELEMENT br EMPTY>
- Elements with only parsed character data
  - <!ELEMENT from (#PCDATA)>
- Elements containing any combination of parsable data
  - <!ELEMENT note ANY>

Elements with Children

- Name of the children elements in parenthesis
  - <!ELEMENT staff (name,phone)>
- Or specified with |
  - <!ELEMENT staff ((name,phone)|(phone,name))>
- Cardinality of the children elements
  - ? zero or one time
  - * zero or more times
  - + one or more times

Attribute Definitions

- Attribute list
- Syntax
  - <!ATTLIST element-name attribute-name attribute-type default-value>
- Example
  - <!ATTLIST payment type CDATA "check">

Attribute Type Value

Type	Description
CDATA	The value is character data
(en1|en2|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

Attribute Default Values

Value	Explanation
value	The default value of the attribute
#REQUIRED	The attribute is required
#IMPLIED	The attribute is not required
#FIXED value	The attribute value is fixed

Entity Definitions

- Define
  - Shortcuts
  - Section of external data
  - Part of a declaration for elements
- Syntax
  - <!ENTITY entity-name "entity-value">
- Program constant

DTD Example

- http://faculty.kutztown.edu/frye/secure/CSC480SW/Examples/productCatalog.xml

XML Schema

- XML Schema Definition (XSD)
- Define structure for XML documents
- XML-based
- Opening tag

<xsd:schema

xmlns:xsd=http://www.w3.org/2000/10/XMLSchema version=”1.0”>

Element Types

- Syntax
  - <element name="….”/>
- Attributes
- Cardinality constraints
  - minOccurs
  - maxOccurs

Attribute Types

- Syntax
  - <attribute name=”…”/>
- Attributes
- Existence
  - use=”x”
  - Values may be optional, required, prohibited, defined value
- Example
  - <attribute name=”id” type=”ID” use=”required”/>

Data Types

- string
- decimal
- integer
- boolean
- date
- time

User-defined Data Types

- Simple data types
  - Cannot use elements or attributes
  - Restrictions on existing data types
- Complex data types
  - Can use elements and attributes
  - Four kinds
    - empty elements
    - elements containing other elements
    - elements containing only text
    - elements containing other elements and text

Complex Data Type Example

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Complex Data Type Example

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid"
                          type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Complex Data Type Example XML

```
<letter>
    Dear Mr.<name>John Smith</name>.
    Your order <orderid>1032</orderid>
    will be shipped on
    <shipdate>2001-07-13</shipdate>.
</letter>
```

Restrictions

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Restriction Example

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions with Patterns

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Processing XML Document

- Process to specify how to display
- Stylesheets
  - CSS
  - XSL (eXtensible Stylesheet Language
- XSL
  - XSLT – XSL Transformations
  - Formatting language